Bandit Algorithms Continued: UCB1

Noel Welsh

09 November 2010

Annoucements

- Lab is busy Wednesday afternoon from 13:00 to 15:00
- (Some) CMUCams are working
- We need rubbish!!!1!
- ROBOtic'10 at Millenium Point on Saturday 27th November 2010, 10:00 - 17:00
  - Free entry
  - Robot competitions in a variety of events
  - http://www.tic.ac.uk/micromouse/ROBOtic10.asp
- Circuit Bending 101 on Saturday November 27th, 11:00 – 5:30
  - £30, payable in advance
  - Meshed Media at Fazeley Studios

Recap

- In the last lecture we raised the issue of learning behaviours, as a way to overcome a limitation of behavioural robotics.
- We looked at the bandit problem as the most basic problem.
- We discussed some basic bandit algorithms, but ran out of time.
- Today we're going to go over the bandit problem again, and this time talk about a simple algorithm that works, UCB1.

The Bandit Problem

- There are K possible actions.
- Each time we take an action we get a reward between 0 and 1.
- There is a different distribution governing rewards for each action.
- Each distribution doesn't change over time (i.e. it is stationary).
- What actions should we take?

Regret

- Bandit algorithms attempt to minimise regret.
- We denote the average (or mean or expected) reward of the best action as μ^* and of any other action j as μ_j. There are a total of K actions. We write $T_j(n)$ for the number of times we have tried action j in a total of n action. Formally, the regret after n actions is defined as

$$regret(n) = \mu^* n - \sum_{j=1}^{K} \mathbb{E}[T_j(n)] \tag{1}$$

Approaches to the Bandit Problem

- Regret is defined in terms of the average reward.
- So if we can estimate average reward we can minimise regret.
- So let's take the action with the highest average reward directly.
  - Assume two actions.
  - Action 1 has reward of 1 with probability 0.3 and otherwise has reward 0f 0.
  - Action 2 has reward of 1 with probability 0.7 and otherwise has reward of 0.
  - Play action 1 first, get reward of 1.
  - Play action 2, get reward of 0.
  - Now average reward of action 1 will never drop to 0, so we'll never play action 2.

Exploring and Exploiting

- This illustrates a classic problem, which is the defining characteristic of *decision making*: the trade-off between exploring and exploiting. Exploring means to try new actions to learn their effects. Exploiting means to try what we know has worked in the past.
- The algorithm above does not explore sufficiently.

Optimism

- The key problem with the algorithm above is that we're too certain of our estimates. When we have seen a single reward of 0 we shouldn't conclude the average reward is 0, but rather than it lies within some *confidence interval* that we adjust to account for the information we have received.
- A confidence interval is a range of values within which we are sure the mean lies with a certain probability. E.g. we could have believe the mean is within $[0.2, 0.5]$ with probability 0.95.
- If we have tried an action less often, our estimated reward is less accurate so the confidence interval is larger. It shrinks as we get more information (i.e. try the action more often).
- Then, instead of trying the action with the highest mean we can try the action with the highest upper bound on its confidence interval.
- This is called an *optimistic* policy. We believe an action is as good as possible given the available evidence.

Chernoff-Hoeffding Bound

- How do we calculate the confidence interval? We can turn to the classic Chernoff-Hoeffding bound to get (most of the way to) an answer.
- Let $X_1, X_2, \ldots, X_n$ be independent random variables in the range $[0, 1]$ with $\mathbb{E}[X_i] = \mu$. Then for $a > 0$,

$$P\left(\frac{1}{n}\sum_{i=1}^{n} X_i \geq \mu + a\right) \leq e^{-2a^2 n} \tag{2}$$

- The other side also holds:

$$P\left(\frac{1}{n}\sum_{i=1}^{n} X_i \leq \mu - a\right) \leq e^{-2a^2 n} \tag{3}$$

- If we wanted to put an upper bound of p on the average reward, we can solve $p = e^{-2a^2 n}$ for a to find out how much we should add. Try this.

Some Insight Into the Chernoff-Hoeffding Bound

- Imagine the rewards are distributed according to a Bernoulli (binary) random variable, as in the example above.
- Then the distribution over n samples (actions) is Binomial.
- The sample average is distributed according $\frac{1}{n}Binomial(n, p)$, where p is the probability of success.
- The standard deviation is then $\sqrt{\frac{p(1-p)}{n}} \leq \frac{1}{2\sqrt{n}}$
- You can see that the Chernoff-Hoeffding bound is closely related to this construction.
- Since the binomial converges to the normal distribution, there is also a strong relationship to the Central Limit Theorem.

UCB1 Overview

- The algorithm UCB1
  [Auer et al.(2002)Auer, Cesa-Bianchi, and Fischer] (for *upper confidence bound*) is an algorithm for the multi-armed bandit that achieves regret that grows only logarithmically with the number of actions taken.
- It is also dead-simple to implement, so good for constrained devices.

UCB1 Details

- For each action j record the average reward $\bar{x}_j$ and number of times we have tried it n_j. We write n for total number of actions we have tried.
- Try the action that maximises $\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$
- That is all! What is the confidence bound we're using?

UCB1's Explore/Exploit Tradeoff

- From out analysis of the Chernoff-Hoeffing bound above we can see that the confidence bound grows with the total number of actions we have taken but shrinks with the number of times we have tried this particular action. This ensures each action is tried infinitely often but still balances exploration and exploitation.

UCB1 Regret Bound

- The regret for UCB1 grows at a rate of ln n. In particular, after n actions it is at most

$$\sum_{j=1}^{K} \frac{4 ln n}{\Delta_j} + \left(1 + \frac{\pi^2}{3}\right) \Delta_j \tag{4}$$

where $\Delta_j = \mu^* - \mu_j$.

UCB1-Tuned

- In practice (but not in theory, 'cause it is too hard to analyse) we can improve on UCB1.
- Note that a Bernoulli random variable with $p = 0.5$ is the reward distribution that will give the highest variance (which is $\frac{1}{4}$).
- We can also compute the sample variance σ_j for each action.
- Then use the upper confidence bound for action j of:

$$\sqrt{\frac{\ln n}{n_j} \min\left(\frac{1}{4}, \left(\sigma_j + \frac{2 \ln n}{n_j}\right)\right)} \tag{5}$$

Extensions

- We can consider observations of the world when making a decision. This is known as the *contextual bandit*.
- There are many variants of the bandit algorithm that address, e.g., costs for switching between arms, or arms with finite lifespan.
- An important variant is the *non-stochastic bandit* which makes no assumptions about the reward distribution (not even identically distributed). This is the Exp3 family of algorithms.
- UCB1 is the building block for tree search algorithms (e.g. UCT) used to, e.g., play games
- Considering the effect of sequence of decisions (i.e. allowing decisions to effect the world) is *reinforcement learning*.

Bibliography

P. Auer, N. Cesa-Bianchi, and P. Fischer.
Finite-time analysis of the multiarmed bandit problem.
Machine learning, 47(2):235–256, 2002.
URL http://www.springerlink.com/index/L7V1647363415H1T.pdf.